Одесский национальный политехнический университет

Кафедра интеллектуальных систем и сетей

Лабораторная работа №8

По предмету «Системное программирование»

На тему «**Межпроцессорное взаимодействие**»

Выполнили

Студенты группы АЕ-145

Михайлов О.М.

Зайцева М.О.

Проверил

Олещук О.В.

Одессы 2017

**Цель лабораторной работы**

Разработать программу с использованием среды разработки Visual Studio или DevCpp.

Протокол должен содержать цель работы, задание, словесное описание, исходный текст программы, результаты работы программы, вывод.

Создать клиент-серверное приложение, где клиентское приложение выполняет интерактивное взаимодействие с пользователем: ввод данных с клавиатуры и вывод результатов на экран. А серверное приложение получает от клиента набор данных для обработки и посылает результат обратно клиенту.

Передача данных между клиентом и сервером должна осуществляться с использованием общей памяти или именованных каналов по выбору студента. Для каждого подключения клиента сервер должен создавать отдельный поток, в котором будет выполняться обработка запроса пользователя.

Передаваемые данные, должны быть размещены в буфере приема/отправки в формате, оптимальном для обработки процессором, т.е. если в задании речь идет об обработке:

* целых чисел - тип данных должен быть целочисленных (int или long);
* вещественных чисел - тип данных float или double;
* строковые значения – тип ASCIIZ-строка или массив символов со счетчиком байт.

Варианты заданий

Вариант 1

Реализовать программу, которая получает массив целых чисел. В этой последовательности необходимо найти простые числа. Найденные простые числа необходимо вывести на экран.

**Ход выполнения работы:**

**package** com.universe.lab8;

**import** java.rmi.Remote;

**import** java.rmi.RemoteException;

**import** javax.swing.JLabel;

**public interface** RemoteService **extends** Remote {

String showNaturalValue(Integer[] number) **throws** RemoteException;

JLabel setText() **throws** RemoteException;

}

**package** com.universe.lab8;

**import** java.math.BigInteger;

**import** java.rmi.Remote;

**import** java.rmi.registry.LocateRegistry;

**import** java.rmi.registry.Registry;

**import** java.rmi.server.UnicastRemoteObject;

**import** javax.swing.JLabel;

**public class** RemoteServiceImpl **implements** RemoteService {

**public static final** String ***BINDING\_NAME*** = **"sample/Service"**;

**public** String **numberToFrame**;

**public** JLabel setText() {

JLabel text = **new** JLabel(**numberToFrame**);

**return** text;

}

**public** String showNaturalValue(Integer[] number) {

StringBuilder result = **new** StringBuilder();

**int** index = 0;

**boolean** status = **false**;

**while** (index < number.**length**) {

BigInteger bigInteger = BigInteger.*valueOf*(number[index]);

**boolean** probablePrime = bigInteger

.isProbablePrime((**int**) Math.*log*(number[1]));*//тест Рабина-Миллера на полноту числа*

**if** (probablePrime) {

result.append(bigInteger + **", "**);

}

index++;

}

**numberToFrame** = result.toString();

**return** result.toString();

}

**public static void** main(String... args) **throws** Exception {

System.***out***.print(**"Starting registry..."**);

**final** Registry registry = LocateRegistry.*createRegistry*(2099); *//регист с ссылками*

System.***out***.println(**" OK"**);

**final** RemoteService service = **new** RemoteServiceImpl();

Remote stub = UnicastRemoteObject.*exportObject*(service, 0);*//rmi сервлет*

System.***out***.print(**"Binding service..."**);

registry.bind(***BINDING\_NAME***, stub);

System.***out***.println(**" OK"**);

**while** (**true**) {

Thread.*sleep*(Integer.***MAX\_VALUE***);*//запущенный поток сервера в ожидание*

}

}

}

**package** com.universe.lab8;

**import** java.awt.GridBagLayout;

**import** java.rmi.registry.LocateRegistry;

**import** java.rmi.registry.Registry;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**public class** ServiceClient {

**public static void** main(String... args) **throws** Exception {

Registry registry = LocateRegistry

.*getRegistry*(**"localhost"**, 2099); *//подключаемся на порт сервера в регистре*

RemoteService service = (RemoteService) registry.lookup(**"sample/Service"**);

Integer[] number = {2, 7, 12, 4, 3, 33, 53, 51, 181};

JFrame frame = **new** JFrame(**"My form"**);

frame.setSize(300, 200);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setLocationRelativeTo(**null**);

frame.setLayout(**new** GridBagLayout());

JLabel textArea = service.setText();

textArea.setHorizontalTextPosition(JLabel.***CENTER***);

textArea.setVerticalTextPosition(JLabel.***BOTTOM***);

System.***out***.println(service.showNaturalValue(number));

textArea = service.setText();

frame.add(textArea);

frame.setVisible(**true**);

}

}

Вывод

В данной лабораторной работе мы разобрали принципы мепроцесорного взаимодействия и создания приложений типа клиент-сервер. Сервер был развёрнут на системе RMI.